

# **MEDITRACK**

Submitted in partial fulfilment of the requirements of

## **PG Diploma in Advanced Computing**

By

**Rupali Pangare 200240320094**  
**Nikhil Deshpande 200240320069**  
**Shubham Nikam 200240320114**  
**Siddhesh Bhavsar 200240320121**  
**Subham Singh 200240320123**

Guide(s):

**Name of the Project Guide**  
Aditya Shirsath

**Name of the Faculty Project Guide**  
Amit Upadhyay



**Centre for Development of Advanced Computing**

**Kharghar**

**February 2020**

# CERTIFICATE

This is to certify that the project entitled “**MediTrack**” is a bonafide work of “**Rupali Pangare** (200240320094), **Nikhil Deshpande** (200240320069), **Shubham Nikam** (200240320114),**Siddhesh Bhavsar**(200240320121),**Subham Singh** (200240320123)etc. submitted to **C-DAC** Mumbai in partial fulfilment of the requirement for the award of the Post Graduate Diploma in Advanced Computing.

**Aditya Shirsath**

**Guide**

**Amit Upadhyay**

**Faculty Guide**

## **DECLARATION**

I declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Rupali Pangare(200240320094)

Nikhil Deshpande(200240320069)

Shubham Nikam(200240320114)

Siddhesh Bhavsar(200240320121)

Shubham Singh(200240320123)

Date:24-01-2021

## **ABSTRACT**

Smart phones are now ubiquitous. The most popular operating system these days is Android. It's easy to use and affordable to all classes. We can find a lot of applications coming up for android based phones. Today's life is much digitalized and fast. Now everything is done at fingertips on our mobile phones so easily. In this busy lifestyle one must not ignore their health. Work goes on but forgetting your medicines is not acceptable. We have come up with the solution to this problem in our project "MediTrack".

MediTrack will be a Web app to help people with their medicine and doctors' routine. It will store all the medications prescribed and help maintain the stock till the time needed. The Purpose of the system is to assist people in reminding them about medicines, tracking their current stock of the required medicines and scheduling appointments at the nearest clinic.

It provides an interface through which user can track the Medicines and can schedule the appointment. Our MediTrack app makes it easy to set up all aspects of user's treatment. Our mission is to simplify your treatment plan no matter how complex it is and to make it possible for you to take control of your own health.

Patient does not need to carry the files of previously held check-up. This application provides a log-in for doctor and patient as well. Doctor can access the medical diagnose information of the patient. Patient is registered with a unique ID. So, the database of history of a patient is available with all the doctors registered with the application. Patient can view their medical diagnose by logging onto their accounts.

## CONTENTS

Chapter	Contents	Page No.
1	INTRODUCTION	08-09
	1.1 Description ( <i>Brief description of project</i> )	08
	1.2 Problem Formulation	08
	1.3 Motivation	08
	1.4 Scope of the project	09
2	REQUIREMENT ANALYSIS	10-11
	2.1 Functional Requirements	10
	2.2 Software Requirement Specification (SRS) ( <i>Hardware and software requirements</i> )	10-11
3	ANALYSIS MODELING	12-18

	<b>3.1 Use-Case Diagrams and description</b>	<b>12</b>
	<b>3.2 Activity Diagrams</b>	<b>13</b>
	<b>3.3 Class Diagram</b>	<b>14-16</b>
	<b>3.4 Sequence Diagram</b>	<b>17-18</b>
<b>4</b>	<b>DESIGN</b>	<b>19-23</b>
	<b>4.1 Data Modeling (<i>E-R Model, Relational tables with its associated Data dictionary</i>)</b> ER Diagram normalized till the third normal form accompanied by the respective data dictionary table should be included	<b>19</b>
	<b>4.2 Architectural Design (<i>Project Flow /architecture with description</i>)</b>	<b>20</b>
	<b>4.2 User Interface Design</b>	<b>21-23</b>
<b>5</b>	<b>IMPLEMENTATION</b>	<b>24-28</b>
	<b>5.1 Implementation Details (<i>code for mentioned users</i>)</b>	<b>24-28</b>
<b>6</b>	<b>TESTING</b>	<b>29-31</b>
	<b>6.1 Test cases (<i>conditions on which testing is done</i>)</b>	<b>29-30</b>
	<b>6.2 Type of Testing used (<i>explanation and reason of testing method used</i>)</b>	<b>30-31</b>
<b>7</b>	<b>CONCLUSIONS &amp; FUTURE SCOPE</b>	<b>32-33</b>

## List of Figures

Fig. No.	Figure Caption	Page No.
1	Use Case Diagram	12
2	Class Diagram	13
3	Activity Diagram	14-16
4	Sequence Diagram	17-18
5	ER Diagram	19
6	User Interface	21-23

# Chapter 1

## Introduction

**MediTrack** is web application which provides an interface through which Admin, Patient, Doctor, Receptionist can register to portal. The application will be used as Medicine and appointment tracking system. The proposed system eliminates the paper work at hospitals and provide an easy solution through which the users would be able to read and update the data on portal.

### 1.1 Description

This project is a web application in which Admin, Doctor, Receptionist and Patient can register to where Receptionist will ask for email id of the patient for registration and patient will get an email having a key generated by the receptionist through which patient can authenticate and can register on the portal. The patient will be able to log in, create and view medical report which comprises of clinic name, disease, doctor name, receptionist name, appointment date etc, which are going under his guidance. The portal will provide information about the patient Medical Report and Appointment with the doctor. This Medical Report will be maintained by Patient, and the Approval/Disapproval of the appointments directly depends on the receptionist /doctor. Patient will get an email regarding the appointment in either of the cases.

### 1.2 Problem Formulation

We often forget about the small things in our day-to-day life because of the fast and busy schedule. But these small things like taking medicines and our weekly or monthly doctor's appointment, they have greater impacts if they are ignored. So, the main objective of building this project is to maintain the health of the user in his busy lifestyle.

The medication should not be ignored and thus MediTrack will help the user by reminding them timely about the medicines and schedule of appointment with the doctor. It also provides the doctor (receptionist) to keep track and schedule of the appointment of the patient.

### 1.3 Motivation



Existing technologies in the market carry out only online delivery of the medicines as per the prescription. Meditrack serves the patient needs by allowing them to take an appointment with the doctor and can keep a track record of their prescriptions, appointments. The Patient will be able to keep track of medicines, Appointment details, stock of required medicines and scheduled reminders at appropriate times for prescribed medicines.

## **1.4 Scope**

This application is used in several Hospitals and Organizations. In the future, Patients can view their medical reports/history. All in one place. The scope of the proposed system is justifiable because a large amount of the population faces the problem of managing the medical records in form of files and papers.

## **Chapter 2**

### **REQUIREMENT ANALYSIS**

#### **2.1 Functional Requirement**

##### **2.1.1 Login of Patient**

- The system will allow the patient to select Clinic Name.
- The system will allow the patient to enter or select Doctor Name from Doctor's List.
- The system will allow the patient to enter Receptionist Name.
- The system will allow the patient to enter the tests.
- The system will allow the patient to take appointment.
- The system will allow the patient to prescriptions.
- The system will allow the patient to view, edit and update the Medical Report.

##### **2.1.2 Login of Receptionist/Doctor**

- The system will allow Receptionist to instantly connect visitors with their host through email.
- The system will allow Receptionist to see the number of patients appointed to a particular doctor.
- The system will allow Receptionist to see the number of appointments.
- The system will allow Receptionist to Confirm/Reject the appointments.

##### **2.1.3 Login of Admin**

- The system will allow the admin view the records.
- The system will allow the admin edit the records.
- The system will allow the admin update the records.
- The system will allow the admin delete the records.

#### **2.2 SOFTWARE REQUIREMENT SPECIFICATIONS(SRS)**

### 2.2.1 Hardware Requirement

- Processor – Intel(R) Core (TM) i7-3770 |CPU @ 3.40GHz
- RAM – 4 GB or above
- Hard Disk – Free disk space of above 6 GB
- Video Monitor (800 × 600 or higher resolution) with at least 256 colors (1024 × 768 High color 16-bit recommended).

### 2.2.2 Software Requirement

- Front End : HTML, Bootstrap, Angular
- Backend: SpringBoot
- Operating System : Windows 7 & 8
- Database: MySQL

### 2.2.3 Tools and Techniques Used

**Eclipse IDE:** The Eclipse platform defines an open architecture so that each plug-in development team can focus on their area of expertise. Let the repository experts build the back ends and the usability experts build the end user tools. If the platform is designed well, significant new features and levels of integration can be added without impact to other tools.

The Eclipse platform uses the model of a common workbench to integrate the tools from the end user's point of view. Tools that you develop can plug into the workbench using well defined hooks called **extension points**.

The platform itself is built in layers of plug-ins, each one defining extensions to the extension points of lower-level plug-ins, and in turn defining their own extension points for further customization. This extension model allows plug-in developers to add a variety of functionality to the basic tooling platform. The artifacts for each tool, such as files and other data, are coordinated by a common platform resource model.

The platform gives the users a common way to work with the tools, and provides integrated management of the resources they create with plug-ins.

Plug-in developers also gain from this architecture. The platform manages the complexity of different runtime environments, such as different operating systems or workgroup server environments. Plug-in developers can focus on their specific task instead of worrying about these integration issues.

**Mysql WorkBench:** MySQL Workbench is a unified visual tool for database architects, developers, and DBAs. MySQL Workbench provides data modeling, SQL development, and comprehensive administration tools for server configuration, user administration, backup, and much more. MySQL Workbench is available on Windows, Linux and Mac OS X.

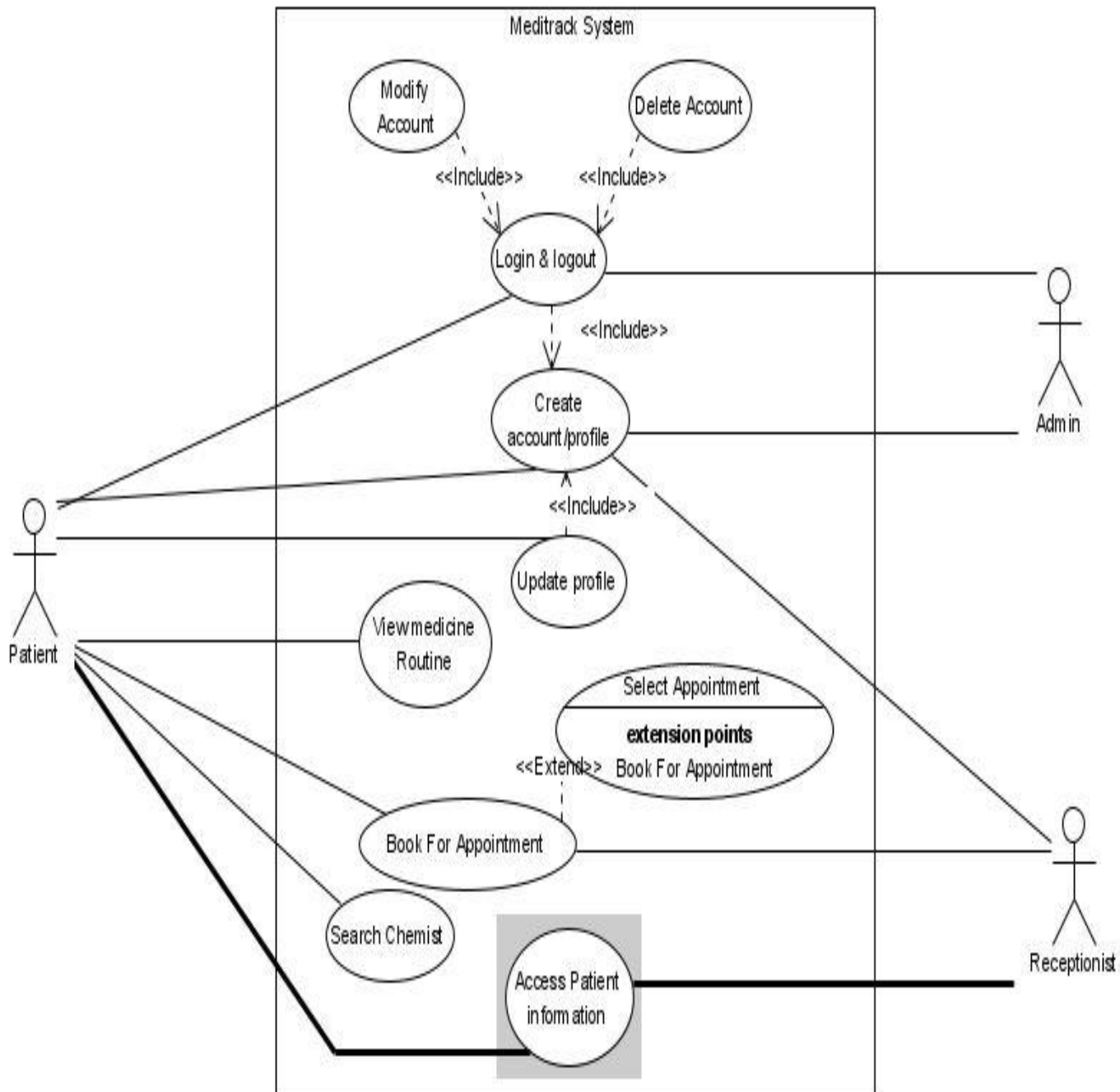
**Visual Studio Code:** Visual Studio Code is a source-code editor that can be used with a variety of programming languages, including Java, JavaScript, Go, Node.js and C++. It is based on

the Electron framework which is used to develop Node.js Web applications that run on the Blink layout engine. Visual Studio Code employs the same editor component (codenamed "Monaco") used in Azure DevOps (formerly called Visual Studio Online and Visual Studio Team Services). Instead of a project system, it allows users to open one or more directories, which can then be saved in workspaces for future reuse. This allows it to operate as a language-agnostic code editor for any language. It supports a number of programming languages and a set of features that differs per language. Unwanted files and folders can be excluded from the project tree via the settings.

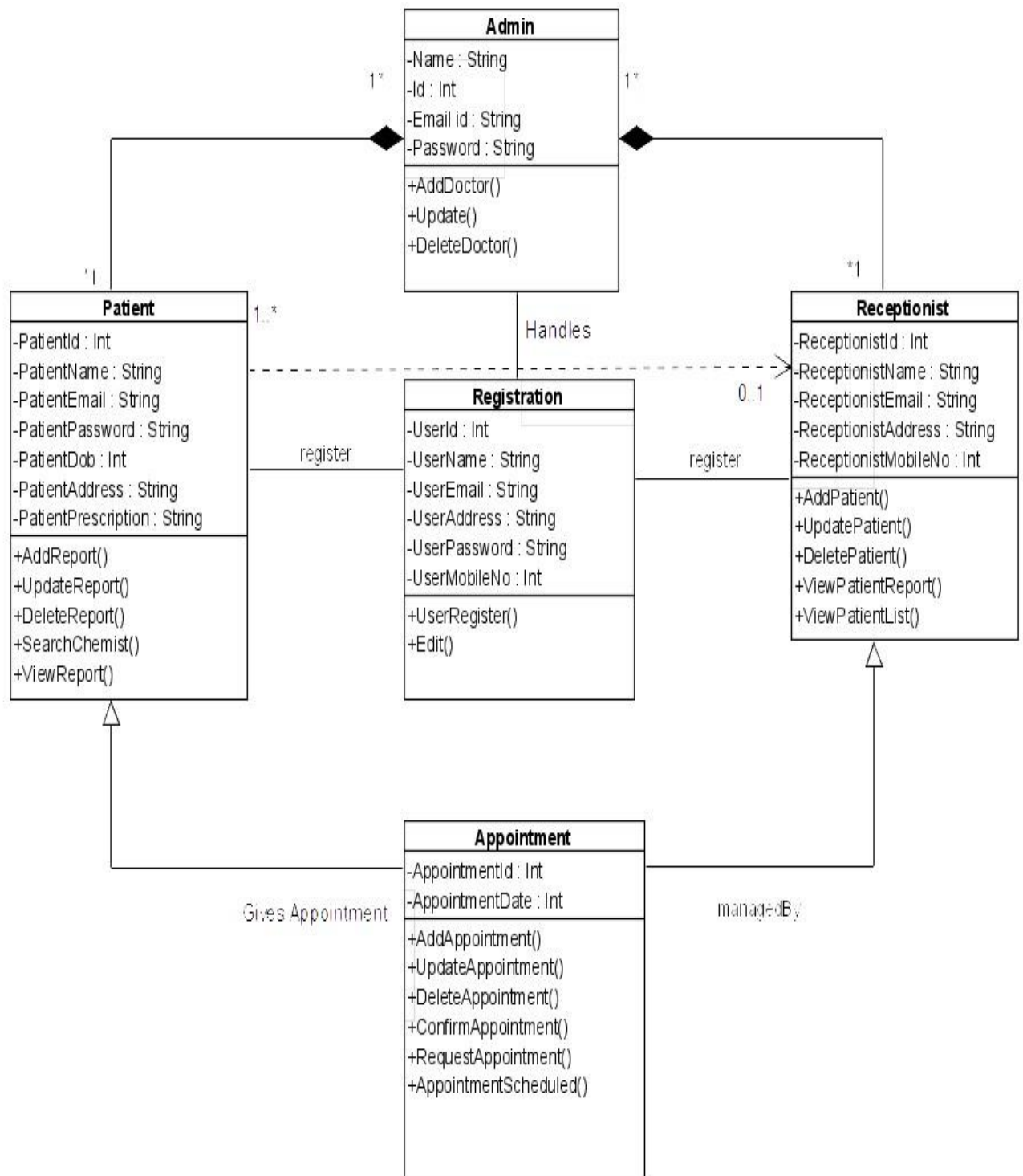
## **Chapter 3**

### **ANALYSIS MODELLING**

### 3.1 Use Case Diagram

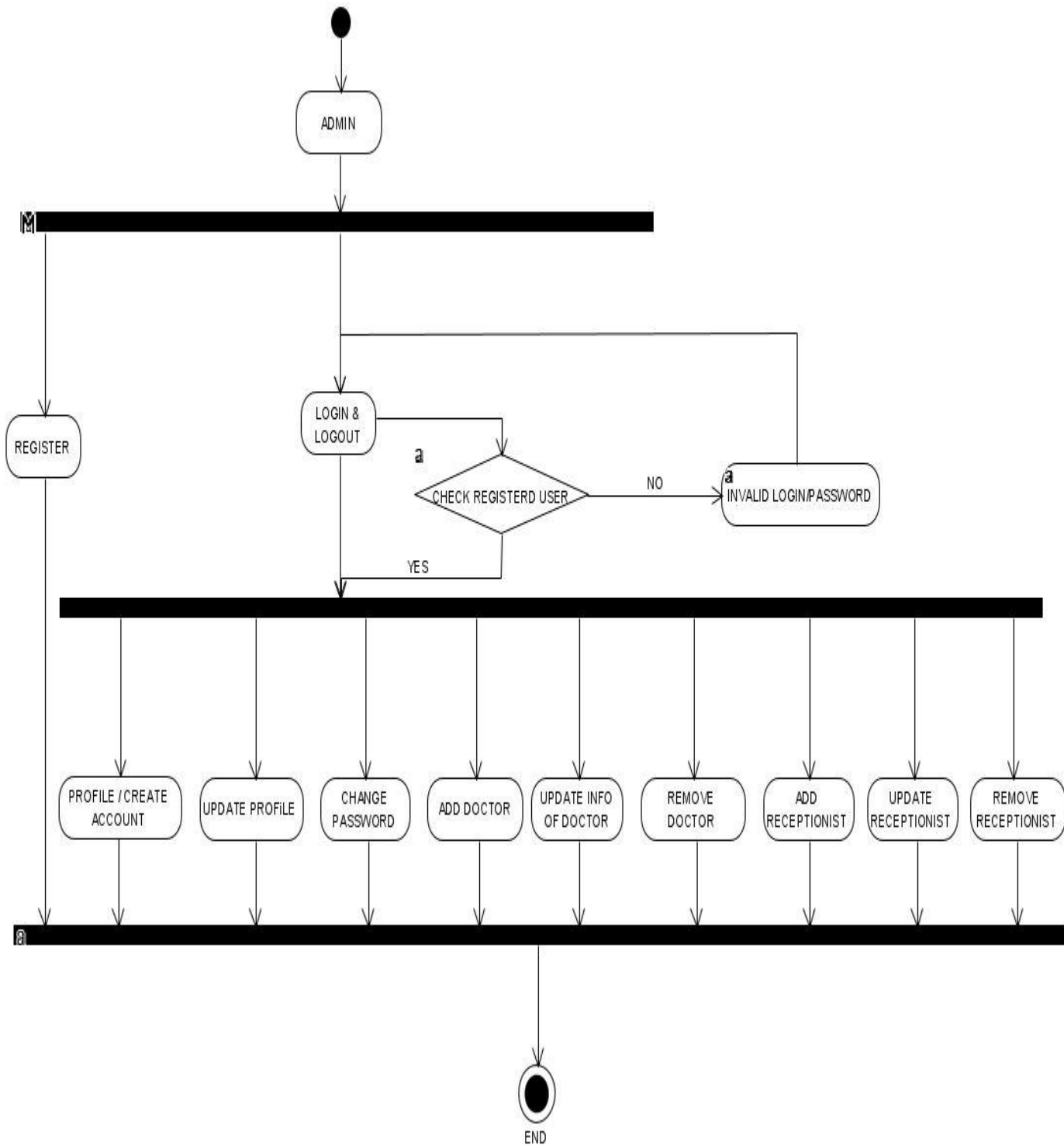


### 3.2 Class Diagram

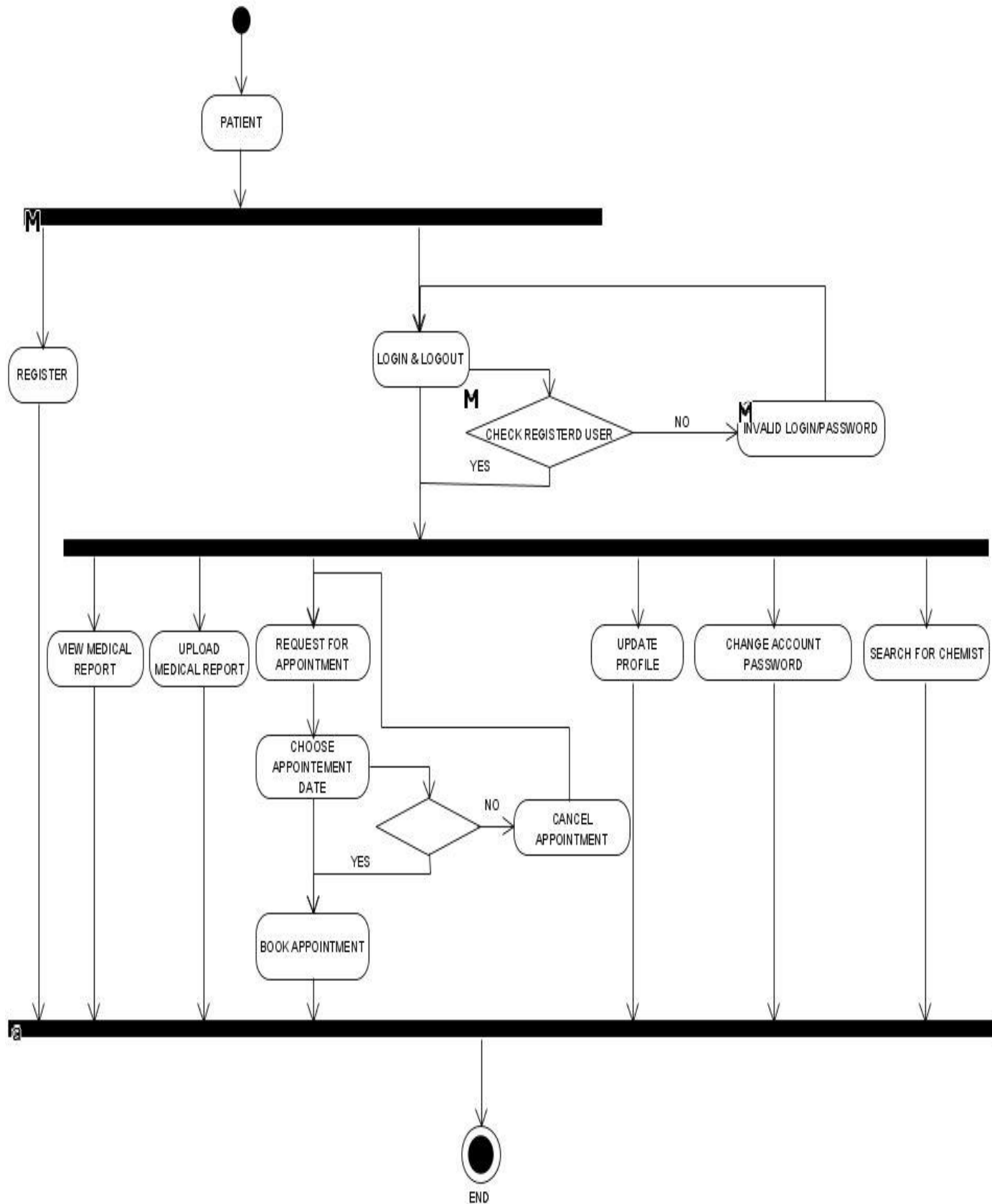


### 3.3 Activity Diagram

Admin:-

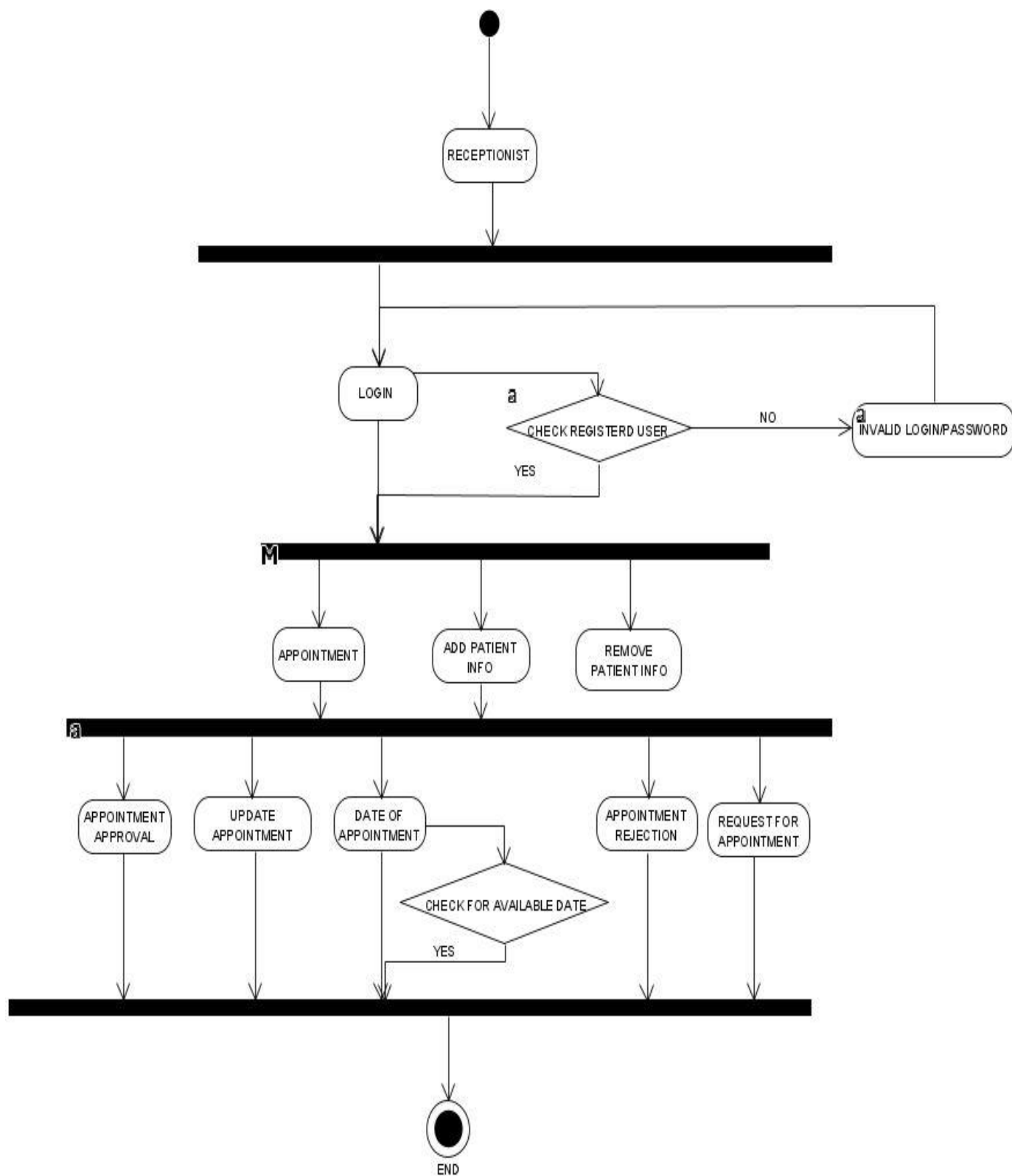


## Patient:-



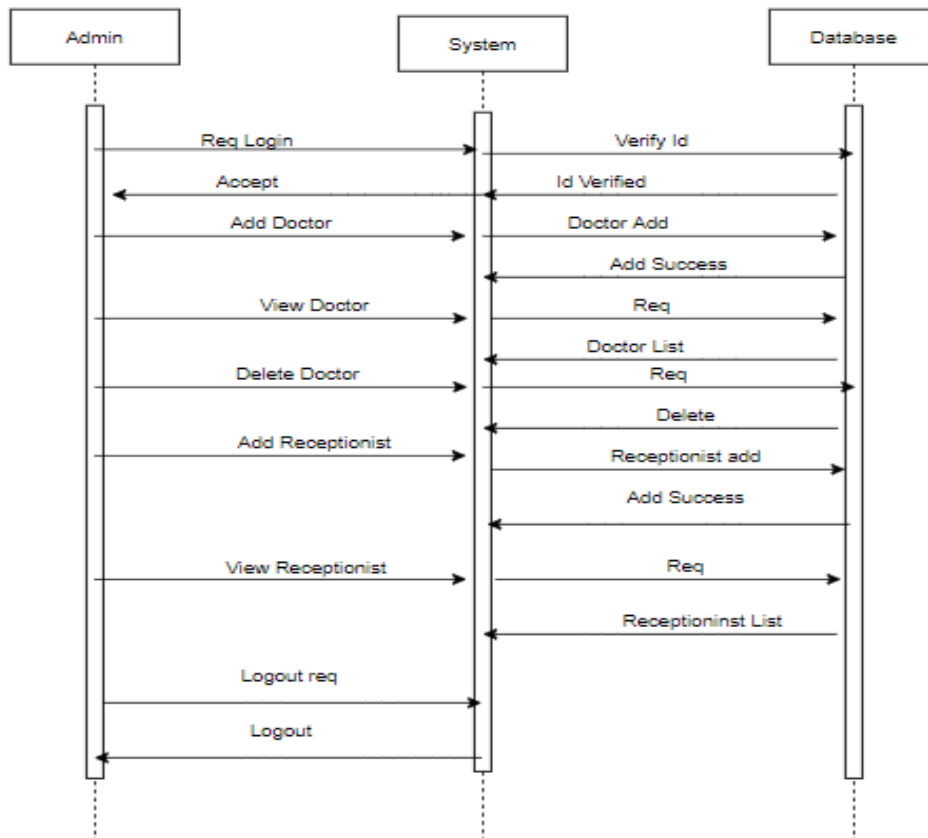


## Receptionist:-

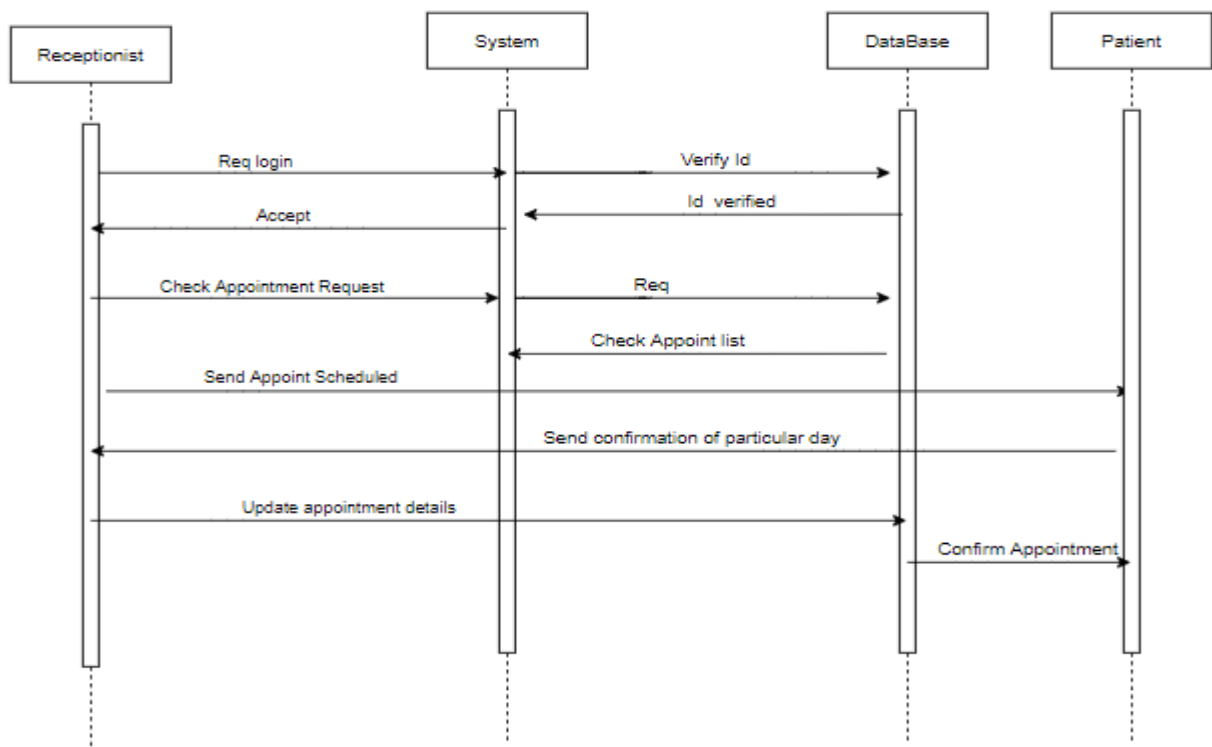


### 3.4 Sequence Diagram

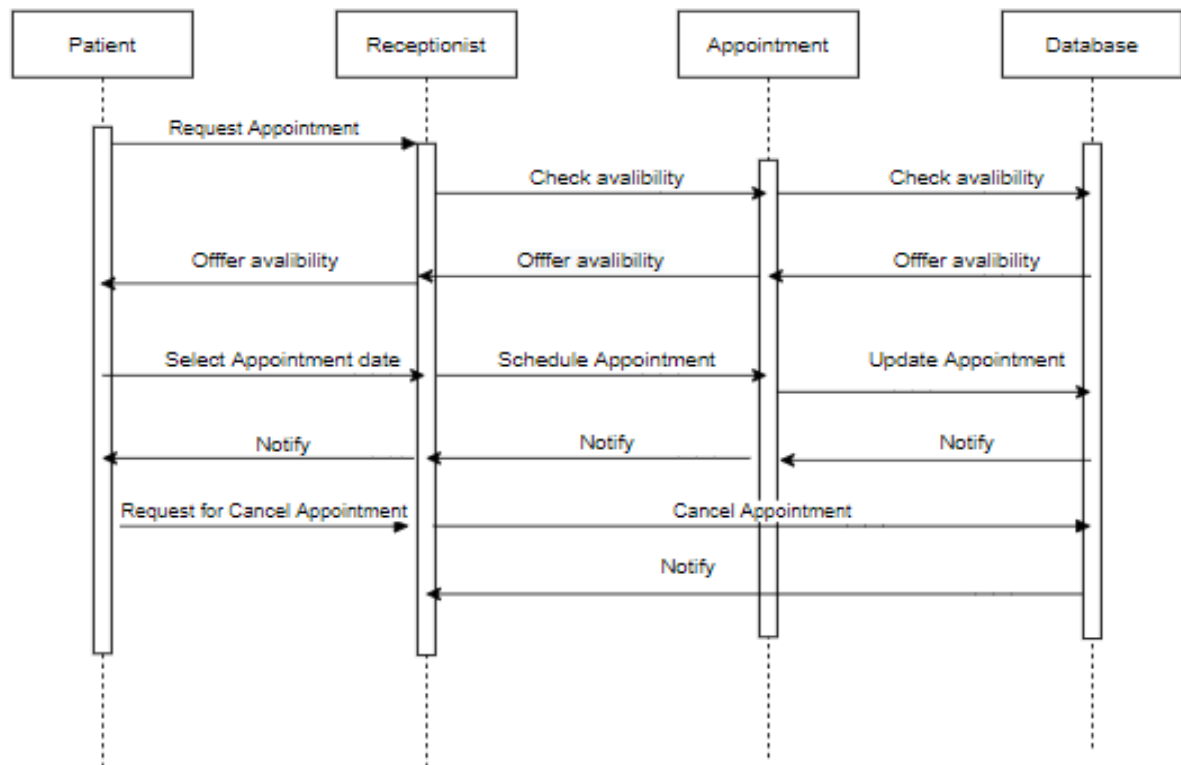
Admin:-



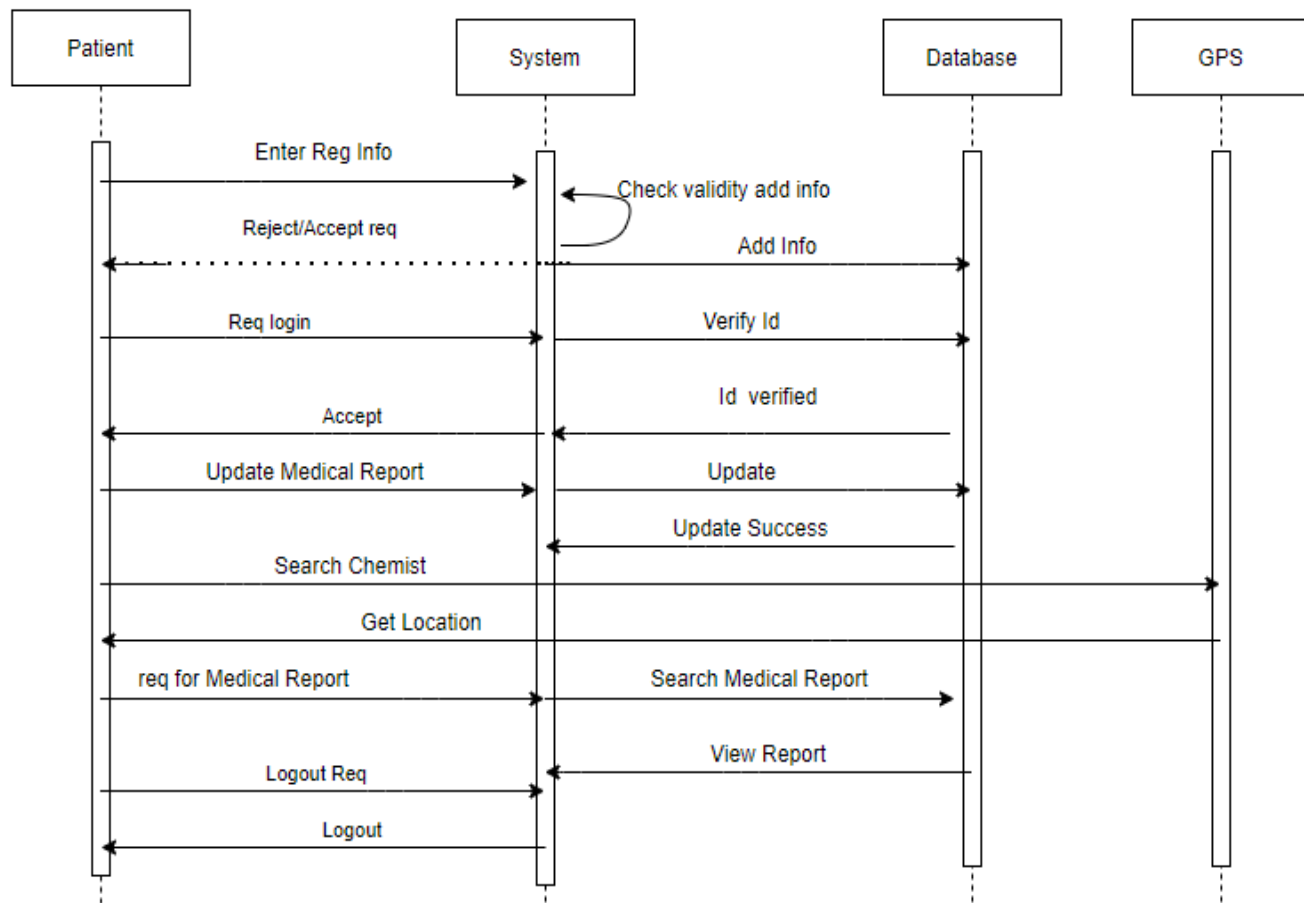
Receptionist:-



## Patient Appointment:-



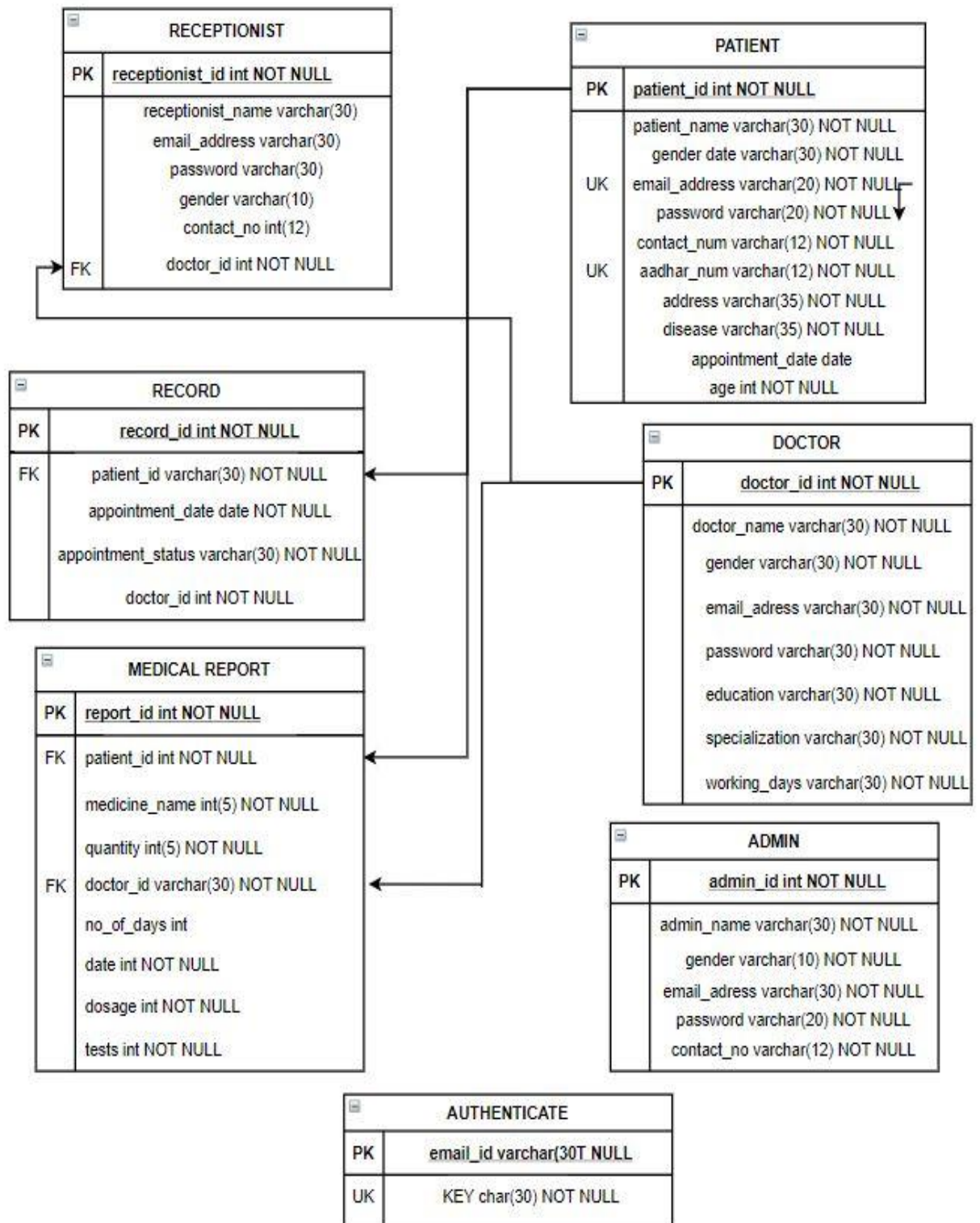
## Patient Registration:-



## Chapter 4

### DESIGN

#### 4.1 Data Modelling (*E-R Model, Relational tables with its associated Data dictionary*)



## 4.2 Architectural Design (*Project Flow /architecture with description*)

### Project flow -

#### 1. Patient -

- Patient go to hospital and get the key from receptionist.
- If Patient is New User then
- He needs to Enter key and Email id and hospital Name for Registration
  
- **Patient\_Registration**-> Patient Enter Details
  
- **Patient Login**
  - If Patient is already Register Then
  - He can Directly login
  - Enter/Update Details of the Medical Report
  - If Patient want to book an appointment -> Request for Appointment->Cancel Appointment.
  - Select Appropriate date of Appointment given by receptionist.
  - Search chemist nearby location.

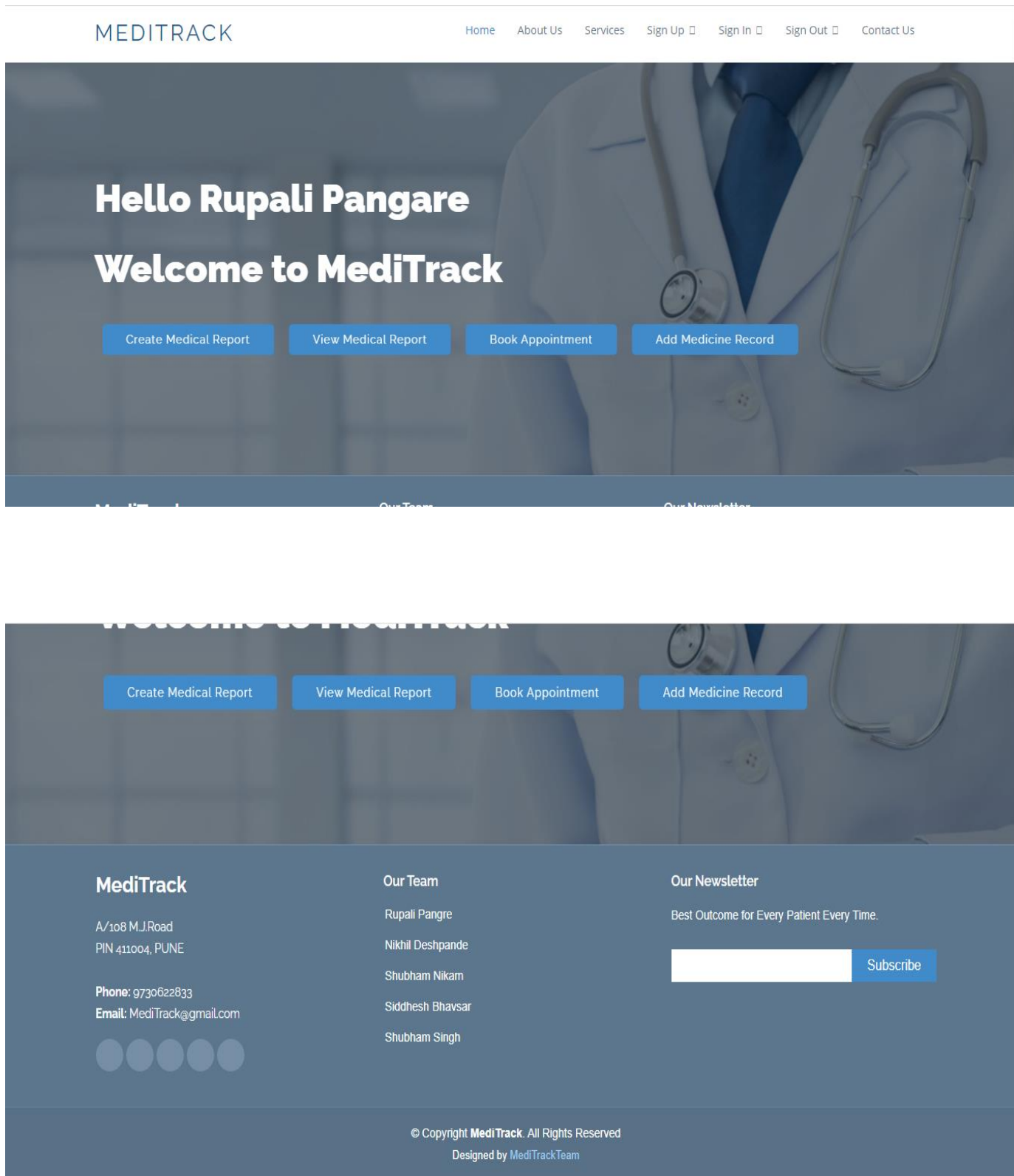
#### 2. Receptionist (Doctor) -

- Receptionist received login credential through admin via E-Mail
- Receptionist Login -> Appointment -> Receptionist check new and pending appointment
- Check for availability of doctor
- Send confirmation or give new appointment date to patient.
- Send remainder of Appointment via Mail.

#### 3. Admin-

- Admin will Login -> Admin will do Registration for Receptionist/doctor and give login credential via E-Mail.

## 4.3 User Interface Design



### Receptionist Register

Name

Email Id

Password

Gender ☐ Male ☒ Female

Contact No

Doctor Id

[Already have an account! Sign In Here](#)

### Login Here!!!

Email

Password

[Forgot Password?](#)  
[Change Password?](#)

## Appointment Details


RECORD ID	RECEPTIONIST Name	PATIENT NAME	APPOINTMENT DATE	STATUS	EDIT
16	sonali	subham	2021-01-29	Requested	<a href="#">Edit</a> <a href="#">Delete</a>
17	sonali	sadicha	2020-01-27	Requested	<a href="#">Edit</a> <a href="#">Delete</a>


## PATIENT SIGNUP

Name	<input type="text" value="Rupali"/>	<input type="text" value="Pangare"/>
	First Name	Last Name
Address	<input type="text" value="Navi Mumbai,Koparkhairne"/>	
Email	<input type="text" value="rupali2997pangare@gmail.com"/>	
New Password	<input type="password" value="*****"/>	
Confirm Password	<input type="password" value="*****"/>	
Mobile Number	<input type="text" value="8976567789"/>	<input type="text" value="22"/>
	Age	
Aadhar No	<input type="text" value="55666766775544"/>	
	Aadhar Number	



## Patient Authentication

 rupali2997pangare@gmail.com

 ...

☐ Remember me

Login

Don't have an account? [Sign Up](#)  
[Forgot your password?](#)

### Medical Report



Patient Name	Rupali Pangare
Clinic Name	Fortis
Disease	Asthama
Doctor Name	Dr.Amit Shukla
Receptionist Name	Anuja
Tests	Kidney Function Test
Appointment Date	2021-01-24

Update Report

Appointment Status

Confirmed



Patient Name

subham

Doctor Name

Dr.Amit Shukla

[Send Mail](#)

## Patient Registration Key



Email Patient Name



Email Id



Key

[Submit](#)

## Chapter 5

### IMPLEMENTATION

#### 5.1 IMPLEMENTATION DETAIL

##### 5.1.1 PATIENT REGISTRATION

```
@CrossOrigin(origins = "http://localhost:4200")
@PostMapping("/PatientDetails")
public Patient createPatient(@RequestBody Patient patient)
{
    return patientRepository.save(patient);
}
```

##### 5.1.2 PATIENT LOGIN

```
@PostMapping("/login")
@ResponseBody
public Patient patientLogin(@RequestBody Patient p, HttpSession Session)
{
    Patient
dt=patientRepository.findByEmailIdAndPassword(p.getEmailId(),p.getPassword() );
    If (Objects.nonNull(dt))
    {
        p.setAadharNo("success");
        return dt;
    }
    else
    {
        p.setAadharNo("fail");
        return p;
    }
}
```

##### 5.1.3 REQUEST FOR APPOINTMENT

```
@PutMapping("/reqForAppointment/{id}")
public AppointmentRecord
reqForAppointments(@PathVariable Long id,@RequestBody AppointmentRecord ar)
{
    ar.setAppointmentStatus("Requested");
}
```

```

        ar.setPatientId(id);

        return appointmentrecordRepository.save(ar);
    }

```

#### 5.1.4 GET MEDICINE DETAILS

```

@CrossOrigin(origins = "http://localhost:4200")

@GetMapping("/report/{id}")

public List<PPrescription> getMedicineDetailsById(@PathVariable int id)

{

    return pprescriptionRepository.findByPid1(id);

}

```

#### 5.1.5 ADD PRESCRIPTION

```

@PutMapping("/addMedicine/{id}")

public PPrescription

addMedicineDetails(@PathVariable Long id,@RequestBody PPrescription pp)

{

    Long l= new Long(id);

    int i=l.intValue();

    pp.setPid1(i);

    return pprescriptionRepository.save(pp);

}

```

#### 5.1.6 SEND PASSWORD VIA MAIL

```

@PostMapping("/sendPasswordByEmail")

public void sendPasswordByEmail(@RequestBody Patient pp)

{

    Patient pp1=patientRepository.findByEmailId(pp.getEmailId());

    System.out.println("tttttttttt");

    System.out.println(pp1.getEmailId());

}

```

```

        System.out.println(pp.getEmailId());

        SimpleMailMessage msg = new SimpleMailMessage();

        System.out.println("TTTTTTTTTT");

        String mail = pp1.getEmailId();

        msg.setTo(mail);

        msg.setSubject("MEDITRACK HEALTH CARE Get Your
Password");

        msg.setText("Hello, "+pp1.getEmailId()+" Your Password is
"+pp1.getPassword()+"\n Your Regards - MEDITRACK HEALTHCARE");

        javaMailSender.send(msg);

        System.out.println("Inside Confirmed loop");

    }

```

### 5.1.7 UPLOAD IMAGE

```

@PostMapping("/uploadPI/{id}")

    public BodyBuilder uplaodImage(@RequestParam("imageFile") MultipartFile
file,@PathVariable Long id) throws IOException

    {

        System.out.println("Original Image Byte Size - " + file.getBytes().length);

        Long l= new Long(id);

        int i=l.intValue();

        PatientImage img = new PatientImage(file.getOriginalFilename(),
file.getContentType(),

            compressBytes(file.getBytes()),i);

        patientimagerepository.save(img);

        return ResponseEntity.status(HttpStatus.OK);

    }

@GetMapping(path = { "/getPI/{id}" })

    public PatientImage getImage(@PathVariable int id) throws IOException {

```

```

        final Optional<PatientImage> retrievedImage =
patientimagerepository.findByPid(id);

        PatientImage img = new PatientImage(retrievedImage.get().getName(),
retrievedImage.get().getType(),

        decompressBytes(retrievedImage.get().getPicByte()),id);

        return img;
    }

    // compress the image bytes before storing it in the database
    public static byte[] compressBytes(byte[] data) {

        Deflater deflater = new Deflater();

        deflater.setInput(data);

        deflater.finish();

        ByteArrayOutputStream outputStream = new ByteArrayOutputStream(data.length);

        byte[] buffer = new byte[1024];

        while (!deflater.finished()) {

            int count = deflater.deflate(buffer);

            outputStream.write(buffer, 0, count);

        }

        try {

            outputStream.close();

        } catch (IOException e) {

        }

        System.out.println("Compressed Image Byte Size - " +
outputStream.toByteArray().length);

        return outputStream.toByteArray();

    }

```

```
// uncompress the image bytes before returning it to the angular application

public static byte[] decompressBytes(byte[] data) {

    Inflater inflater = new Inflater();

    inflater.setInput(data);

    ByteArrayOutputStream outputStream = new ByteArrayOutputStream(data.length);

    byte[] buffer = new byte[1024];

    try {

        while (!inflater.finished()) {

            int count = inflater.inflate(buffer);

            outputStream.write(buffer, 0, count);

        }

        outputStream.close();

    } catch (IOException ioe) {

    } catch (DataFormatException e) {

    }

    return outputStream.toByteArray();

}
```

## Chapter 6

### TESTING

#### 6.1 Test cases (*conditions on which testing is done*)

Test Id	Item to be Tested	Steps	Input	Actual Output	Expected Output	Pass/Fail
1	User Id	User enters user Id	User Id	Display Success	Display Message successful	Pass
2	System checks for proper username and password entered by users	System compares the data entered by user and the entered data in database				
		If username and password is valid		Make Connection	Make connection	Pass
		If username and password is invalid		Report invalid user id	Report error	Pass



<b>3</b>	<b>System checks whether details of user are entered as per the format</b>	<b>System checks the data entered by user is in valid form or not.</b>				
		<b>If valid</b>	<b>User entered data</b>	<b>Entered in database</b>	<b>Entered in database</b>	<b>Pass</b>
		<b>If invalid</b>	<b>User entered data</b>	<b>“Invalid Data” message will be printed</b>	<b>“Invalid Data” message will be printed</b>	<b>Pass</b>

## 6.2 Type of Testing used

### Introduction

- The aim of testing process is to identify all defects in a software product. Testing is any activity aimed at evaluating the software for quality results it produces and the quality of results it can handle. Testing is an operation to detect the differences between the expected (required) result and the actual result.
- Testing a program consists of subjecting the program to a test inputs or test cases and observing if the program behaves as expected. If the program fails to behave as expected, then the condition under which failures occurs are noted for later debugging and correction.
- Our goal is to design a series of test cases that would have a high likelihood of finding errors. The software testing techniques provide a systematic guidance for designing tests that exercise the internal logic of software components and exercise the input & output domains of the program to uncover errors in program function, behaviour and performance.
- Software is tested from two different perspectives:
  - (1) Internal program logic is exercised using “White Box” test case design techniques.
  - (2) Software requirements are exercised using “Black Box” test case design techniques. In both cases, the intent is to find maximum number of errors with minimum effort and time.

### **6.2.1 System test objective and scope**

The main aim to test this is to insure that:

- The Proposed system permits only secure and authenticate access.
- Thus, requires the user to enter the URL in correct format.
- Does all validation time to time as per need.
- Takes a single input as user id for the detection of anomalies that is used to generate the recommendations.
- Appropriates alerts are generated as per the condition for user convenience.
- Database is updated time to time as the user transaction process proceeds.

## **Chapter 7**

### **Conclusions**

Patient records are the primary repository of data in the information-intensive health care industry. Although clinical information is increasingly likely to be computerized, the current, predominant mode for recording patient care data remains the paper record. Paper records have the advantages of being familiar to users and portable; when they are not too large, users can readily browse through them. Paper records, however, have serious, overriding limitations that frequently frustrate users and perpetuate inefficiencies in the health care system. Further, the impact of these limitations is growing as the health care system becomes more complex. Modern patient care requirements have outgrown the paper record.

Quality improvement and cost containment continue to be major concerns for the health care industry. Quality assurance; utilization management; appropriateness, effectiveness, and outcomes assessment; clinical practice guidelines; and value purchasing are all prominent responses to the quality or cost challenges faced by present-day health care. Each of these initiatives increases the legitimate demand for complete, accurate, readily accessible patient data.

Health care professionals today face an unprecedented information explosion as the quantity and complexity of patient data and medical knowledge increase practically daily. Current patient records cannot adequately manage all the information needed for patient care. Paper patient records have not kept and cannot keep pace with the rapidly changing health care system. As a result, they increasingly impede effective decision-making throughout the health care sector—from the bedside to the formulation of national health care policy.

## REFERENCES

1. <https://stackoverflow.com/questions/34798744/understanding-spring-boot>
2. <https://zetcode.com>
3. <https://stackabuse.com/spring-security-forgot-password-functionality/>
4. [www.youtube.com](http://www.youtube.com)
5. <http://www.telusko.com/>
6. Spring Boot in Action By Walls Craig
7. Angular Essentials By Kumar Dhananjay
8. Database Management System by Paul

